Simple Linear Regression

In [166]:
```
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt

% matplotlib inline
```

In [167]:
```
iris     = sns.load_dataset('iris')
nsamples = 5
iris.sample(nsamples)
```

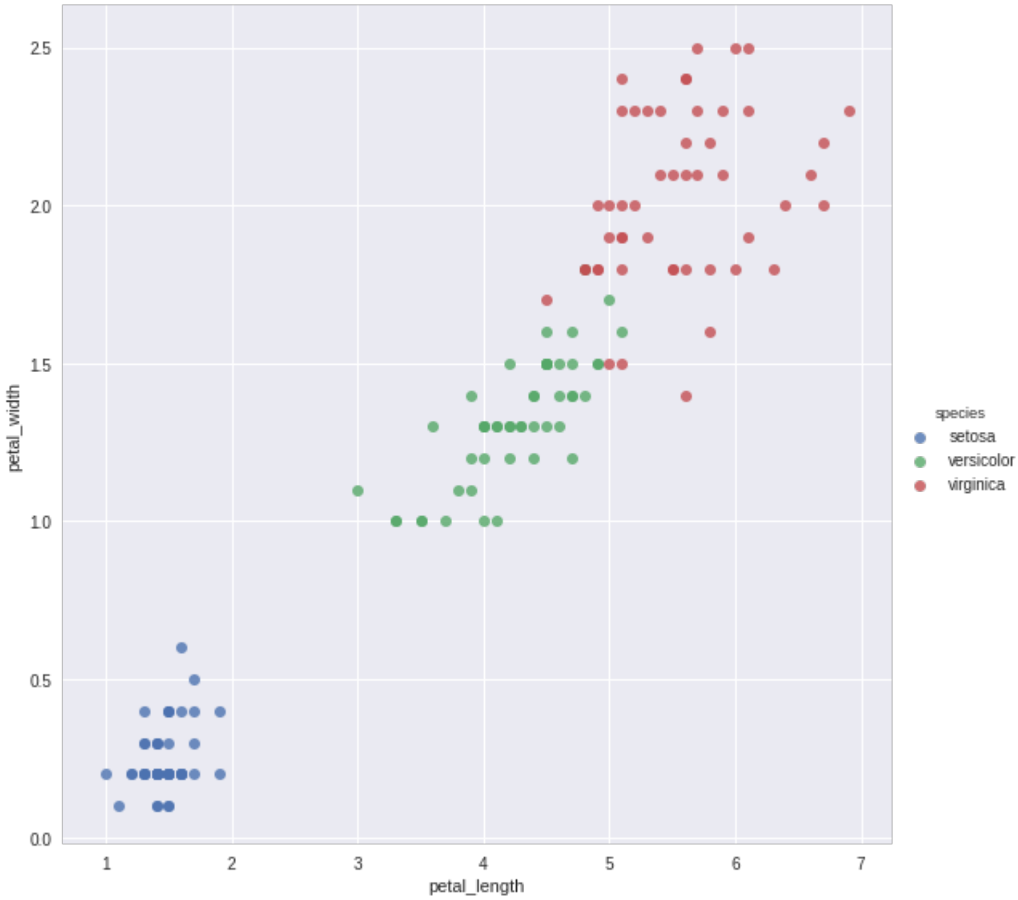
Out[167]:

	sepal_length	sepal_width	petal_length	petal_width	species
49	5.0	3.3	1.4	0.2	setosa
141	6.9	3.1	5.1	2.3	virginica
63	6.1	2.9	4.7	1.4	versicolor
94	5.6	2.7	4.2	1.3	versicolor
143	6.8	3.2	5.9	2.3	virginica

In [168]:
```
column = { 'feature': 'petal_length', 'target': 'petal_width' }
```

In [169]:
```
sns.lmplot(x = column['feature'], y = column['target'], data = iris, fit_r
eg = False, hue = 'species', size = 8)
```

Out[169]: <seaborn.axisgrid.FacetGrid at 0x7f9e9f795b38>



In [170]:
```
x, y = iris[column['feature']], iris[column['target']]
```

In [171]:
```
from scipy.stats import pearsonr
```

In [172]:
```
r, p = pearsonr(x, y)
r
```

Out[172]: 0.96286543140279612

In [173]:
```
def predict(x, beta0, beta1):
    return beta0 + beta1 * x
```

In [200]:
```
def plot(beta0, beta1):
    sns.lmplot(x = column['feature'], y = column['target'], data = iris, f
it_reg = False, hue = 'species', size = 8)
    plt.plot(x, predict(x, beta0, beta1))
```

In [201]:
```
beta0, beta1 = np.random.rand(2)
beta0, beta1
```

Out[201]: (0.94634819241045165, 0.23293986319851268)

In [202]:
```
from prettytable import PrettyTable
from IPython.core.display import display, HTML
```

In [203]:

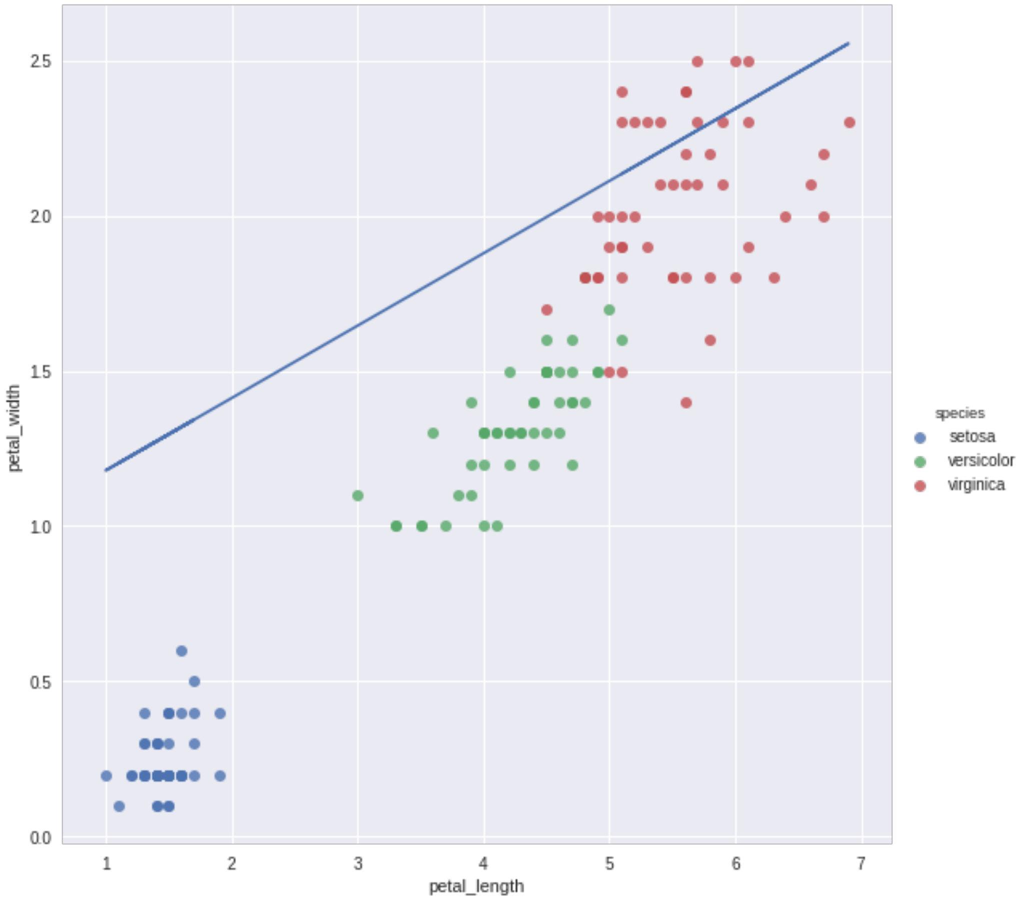
```
def display_table(beta0, beta1):
    table = PrettyTable(['$i$',
                         '$x_{i}$',
                         '$y_{i}$',
                         '$\hat{y}_{i}$',
                         '$(y_{i}-\hat{y}_{i})$',
                         '$(y_{i}-\hat{y}_{i})^{2}$'])
    size  = len(x)
    m     = 0
    rows  = [ ]

    for i in range(size):
        p = predict(x[i], beta0, beta1)
        e = y[i] - p
        s = np.power(e, 2)
        m = m + s

        rows.append([i + 1, x[i], y[i], p, e, s])

    for i in range(10):
        table.add_row(rows[i])

    table.add_row(['', '', '', '', '', m])

    display(HTML(table.get_html_string()))
```

In [204]:

```
display_table(beta0, beta1)
```

i	x_i	y_i	$\hat{y}_i$	$(y_i - \hat{y}_i)$	$(y_i - \hat{y}_i)^2$
1	1.4	0.2	1.27246400089	-1.07246400089	1.1501790332
2	1.4	0.2	1.27246400089	-1.07246400089	1.1501790332
3	1.3	0.2	1.24917001457	-1.04917001457	1.10075771947
4	1.5	0.2	1.29575798721	-1.09575798721	1.20068556653
5	1.4	0.2	1.27246400089	-1.07246400089	1.1501790332
6	1.7	0.4	1.34234595985	-0.942345959848	0.888015908042
7	1.4	0.3	1.27246400089	-0.972464000888	0.945686233024
8	1.5	0.2	1.29575798721	-1.09575798721	1.20068556653
9	1.4	0.2	1.27246400089	-1.07246400089	1.1501790332
10	1.5	0.1	1.29575798721	-1.19575798721	1.42983716397
					79.9363589781

In [205]:
```
plot(beta0, beta1)
```



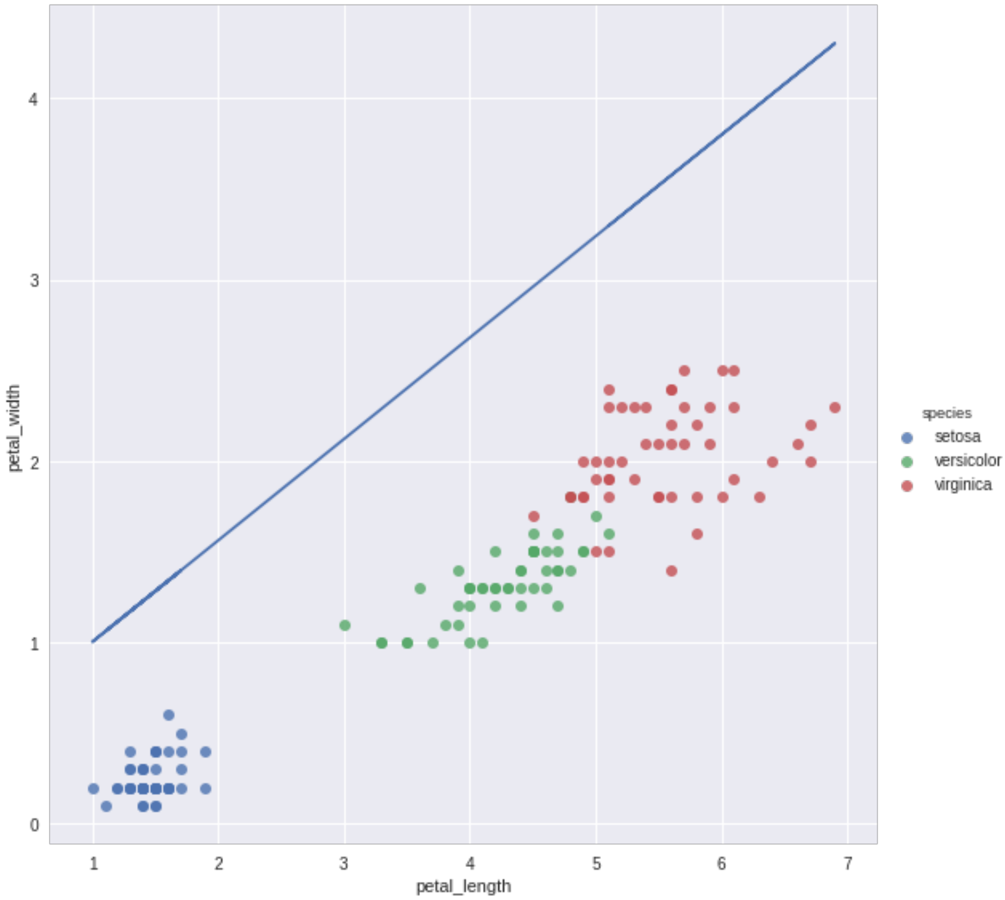
In [206]:
```
beta0, beta1 = np.random.rand(2)
beta0, beta1
```

Out[206]: (0.44395852212752296, 0.55909523711243136)

In [207]:
```
display_table(beta0, beta1)
```

i	x_i	y_i	$\hat{y}_i$	$(y_i - \hat{y}_i)$	$(y_i - \hat{y}_i)^2$
1	1.4	0.2	1.22669185408	-1.02669185408	1.05409616324
2	1.4	0.2	1.22669185408	-1.02669185408	1.05409616324
3	1.3	0.2	1.17078233037	-0.970782330374	0.942418332966
4	1.5	0.2	1.2826013778	-1.0826013778	1.17202574321
5	1.4	0.2	1.22669185408	-1.02669185408	1.05409616324
6	1.7	0.4	1.39442042522	-0.994420425219	0.988871982092
7	1.4	0.3	1.22669185408	-0.926691854085	0.858757792427
8	1.5	0.2	1.2826013778	-1.0826013778	1.17202574321
9	1.4	0.2	1.22669185408	-1.02669185408	1.05409616324
10	1.5	0.1	1.2826013778	-1.1826013778	1.39854601877
					287.488595635

In [208]:
```
plot(beta0, beta1)
```



In [209]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x.values.reshape((len(x), 1)), y.values.reshape((len(y), 1)))
```

Out[209]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=Fals
e)
```

In [210]:
```
beta0 = lr.intercept_[0]
beta1 = lr.coef_[0,0]
beta0, beta1
```

Out[210]:
```
(-0.36307552131902909, 0.41575541635241153)
```

In [211]:
```
display_table(beta0, beta1)
```

i	x_i	y_i	$\hat{y}_i$	$(y_i - \hat{y}_i)$	$(y_i - \hat{y}_i)^2$
1	1.4	0.2	0.218982061574	-0.0189820615743	0.000360318661612
2	1.4	0.2	0.218982061574	-0.0189820615743	0.000360318661612
3	1.3	0.2	0.177406519939	0.0225934800609	0.000510465341262
4	1.5	0.2	0.26055760321	-0.0605576032096	0.00366722330649
5	1.4	0.2	0.218982061574	-0.0189820615743	0.000360318661612
6	1.7	0.4	0.34370868648	0.0562913135199	0.0031687119778
7	1.4	0.3	0.218982061574	0.0810179384257	0.00656390634674
8	1.5	0.2	0.26055760321	-0.0605576032096	0.00366722330649
9	1.4	0.2	0.218982061574	-0.0189820615743	0.000360318661612
10	1.5	0.1	0.26055760321	-0.16055760321	0.0257787439484
					6.31009637925

In [247]:
```
sns.lmplot(x = column['feature'], y = column['target'], data = iris, fit_r
eg = False, hue = 'species', size = 8)
plt.plot(x, predict(x, beta0, beta1))
```

Out[247]: [<matplotlib.lines.Line2D at 0x7f9e9f331710>]